Deciding Parity Games in Quasipolynomial Time*

Cristian S. Calude[1], Sanjay Jain[2], Bakhadyr Khoussainov[1], Wei Li[3] and Frank Stephan[2,3]

[1] Department of Computer Science, University of Auckland
Private Bag 92019, Auckland, New Zealand
{cristian,bmk}@cs.auckland.ac.nz
[2] Department of Computer Science, National University of Singapore
13 Computing Drive, COM1, Singapore 117417, Republic of Singapore
{sanjay,fstephan}@comp.nus.edu.sg
[3] Department of Mathematics, National University of Singapore
10 Lower Kent Ridge Road, S17, Singapore 119076, Republic of Singapore
liwei.sg@gmail.com

Abstract It is shown that the parity game can be solved in quasipolynomial time. The parameterised parity game – with n nodes and m distinct values (aka colours or priorities) – is proven to be in the class of fixed parameter tractable (**FPT**) problems when parameterised over m. Both results improve known bounds, from runtime $n^{O(\sqrt{n})}$ to $O(n^{\log(m)+6})$ and from an **XP**-algorithm with runtime $O(n^{\Theta(m)})$ for fixed parameter m to an **FPT**-algorithm with runtime $O(n^5)+g(m)$, for some function g depending on m only. As an application it is proven that coloured Muller games with n nodes and m colours can be decided in time $O((m^m \cdot n)^5)$; it is also shown that this bound cannot be improved to $O((2^m \cdot n)^c)$, for any c, unless $\mathbf{FPT} = \mathbf{W[1]}$.

1 Introduction

A parity game is given by a directed graph (V, E), a starting node $s \in V$, a function *val* which attaches to each $v \in V$ an integer value (also called colour) from a set $\{1, 2, \ldots, m\}$; the main parameter of the game is n, the number of nodes, and the second parameter is m. Two players Anke and Boris move alternately in the graph with Anke moving first. A move from a node v to another node w is valid if (v, w) is an edge in the graph;

* S. Jain is supported in part by NUS grant C252-000-087-001; B. Khoussainov is supported in part by the Marsden Fund grant of the Royal Society of New Zealand; Furthermore, S. Jain, B. Khoussainov and F. Stephan are supported in part by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2016-T2-1-019 / R146-000-234-112. This work is accepted for the conference STOC 2017 [9].

furthermore, it is required that from every node one can make at least one valid move. Anke and Boris own certain values. The alternate moves by Anke and Boris and Anke and Boris and ... define an infinite sequence of nodes which is called a play. The values of the nodes can always be chosen such that one player wins at the even numbers and the other player wins at the odd numbers. Anke wins a play through nodes $a_0, a_1, \ldots$ iff the limit superior (that is, the largest value appearing infinitely often) of the sequence $val(a_0), val(a_1), \ldots$ is a number she owns, that is, a number of her parity. An example is the game as given in Figure 1; here the nodes

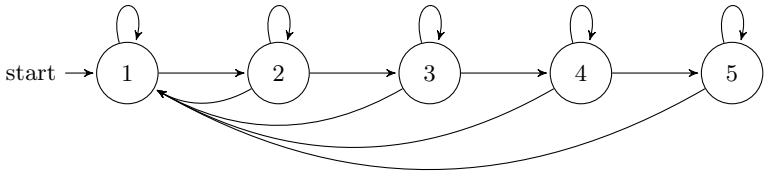

Figure 1.

are labeled with their values, which are unique (but this is not obligatory); furthermore, Anke has even and Boris has odd parity. Boris has now the following memoryless (that is, moves are independent of the history) winning strategy for this game: $1 \to 1$, $2 \to 3$, $3 \to 3$, $4 \to 5$, $5 \to 5$. Whenever the play leaves node 1 and Anke moves to node 2, then Boris will move to node 3. In case Anke moves to node 4, Boris will move to node 5. Hence, whenever the play is in a node with even value (this only happens after Anke moved it there), in the next step the play will go into a node with a higher odd value. So the largest infinitely often visited node value is odd and therefore the limit superior of these numbers is an odd number which justifies Boris' win. Hence Boris has a winning strategy for the parity game given above.

For parity games, in general, the winner can always use a *memoryless winning strategy* [5,28,48,49,64]. This fact will be one central point in the results obtained in this paper: the parity game will be augmented with a special statistics – using polylogarithmic space – which indicates the winner correctly after a finite time whenever the winner employs a memoryless winning strategy.

It should be pointed out that one can also consider games where it only depends on the node which player moves and the players do not ne-

cessarily take turns. Both versions of parity or Muller games can be translated into each other with a potential increase of the number of nodes by a factor 2. In the case that one goes from turn-based to position-based Muller games, one doubles up each node: Instead of the node v one uses a node (Anke, v), when it is Anke's turn to move, and a node (Boris, v), when it is Boris' turn to move. For the other direction, each node w receives a prenode w' with exactly one outgoing edge from w' to w. Now, for each edge (v, w) from the original game, if the same player moves at v and at w in the original game, then one puts the edge (v, w') into the new game, else one puts the edge (v, w) into the new game. The rationale behind this is that the longer path $v - w' - w$ has even length in the case that the players moving at v and w should be the same for alternating moves. Furthermore, if Anke moves at the original starting node s, then s is also the starting node of the new game, else s' is the starting node of the new game.

Parity games are a natural class of games which are not only interesting in their own right, but which are also connected to fundamental notions like μ-calculus, modal logics, tree automata and Muller games [3,6,7,14,27,28,38,57,61,62,63]. A possible application of good algorithms to solve parity games would be that one gets better algorithms to decide the theory of certain tree automatic structures [23,24,47] and to employ such algorithms in order to understand these structures better.

For investigating the complexity side of the game, it is assumed that the game is given by a description in size polynomial in the number n of nodes and that one can evaluate all relevant parts of the description in logarithmic space. A possibility is to store the following three items for each game (where Anke moves first and starts from node 1):

- two numbers m, n with $1 \leq m \leq n$ and one bit which says whether the values owned by player Anke are the even or the odd numbers;
- the game graph given by a table, that is, for each pair of nodes, a bit which says whether there is a directed edge between the two nodes or not;
- the values of the nodes given by another table which holds, for each node, a binary number from $\{1, 2, \ldots, m\}$.

An important open problem for parity games is the *time complexity* for finding the winner of a parity game, when both players play optimally; the first algorithms took exponential time [48,64] and subsequent studies searched for better algorithms [40,42,44,51,56,57,58]. Many researchers, including Emerson and Jutla [28] in 1991, asked whether the winner of a

parity game can be determined in polynomial time.

Emerson, Jutla and Sistla [29] showed that the problem is in **NP** $\cap$ **co-NP** and Jurdzinski [41] improved this bound to **UP** $\cap$ **co-UP**. This indicates that the problem is not likely to be hard for **NP** and might be solvable faster than in exponential time. Indeed, Petersson and Vorobyov [51] devised a subexponential randomised algorithm and Jurdzinski, Paterson and Zwick [44] a deterministic algorithm of similar complexity (more precisely, the subexponential complexity was approximately $n^{O(\sqrt{n})}$). Besides this main result, there are also various practical approaches to solve special cases [3,19,32] or to test out and analyse heuristics [10,35,42]; however, when Friedmann and Lange [30] compared the various parity solving algorithms from the practical side, they found that Zielonka's recursive algorithm [64] was still the most useful one in practice. McNaughton [48] showed that the winner of a parity game can be determined in time $n^{m+O(1)}$ and this was subsequently improved to $n^{m/2+O(1)}$ [8,59] and to $n^{m/3+O(1)}$ [56,58], where n is the number of nodes and m is the maximum value (aka colour aka priority) of the nodes.

The consideration of the parameter m is quite important; it is a natural ingredient of the parity games. Schewe [57,58] argues that for many applications which are solved using parity games, the parameter m is much smaller than n, often by an exponential gap. For example, when translating coloured Muller games into parity games in the way done by McNaughton [48] and Björklund, Sandberg and Vorobyov [4], the number of values is, for all but finitely many games, bounded by the logarithm of the number of nodes, see the proof of Theorem 13 below. Similarly, an important application of parity games is the area of reactive synthesis. Here one translates LTL-formulas into a Büchi automaton which needs to be determinised by translating it into a parity automaton. Building on work of Safra [54,55], Piterman [52] showed that he can translate nondeterministic Büchi automata with n states into parity automata with $2 \cdot n^n \cdot n!$ states and $2n$ values. In other words, he can evaluate various conditions on these parity automata by determining the winner in the corresponding parity game. Also Di Stasio, Murano, Perelli and Vardi [18] investigated in their experiments various scenarios where the number m is logarithmic in n. The present work takes therefore the parameter m into consideration and improves the time bounds bounds in two ways:

- The overall time complexity is $O(n^{\lceil \log(m) \rceil + 6})$ which provides a quasi-polynomial bound on the runtime, as one can always choose $m \leq n$.
- Furthermore, if $m < \log(n)$, then the overall time complexity is $O(n^5)$, which shows that the problem is fixed parameter tractable when para-

meterised by m; the parity games are therefore in the lowest time complexity class usually considered in parameterised complexity.

Prior investigations have already established that various other parameterisations of parity games are fixed-parameter tractable, but the parameterisation by m was left open until now. Chatterjee [12] pointed out to the authors that one can also write the result in a product form with Parity Games be solvable in time $O(2^m \cdot n^4)$ for all m, n; the proof uses just the methods of Theorem 8 but keeping m as a parameter and not using explicitly the bound of $m \leq \log(n)$ which, when invoked into above formula, would give the bound $O(n^5)$.

An application of the results is that coloured Muller games with n nodes and m colours can be decided in time $O((m^m \cdot n)^5)$; Theorem 14 and Corollary 15 below show that this bound cannot be improved to $O((2^m \cdot n)^c)$ for any c unless $\mathbf{FPT} = \mathbf{W[1]}$.

Subsequent research [22,33,43,60] has provided the additional runtime bound

$$O(\lceil m/\log(n)\rceil^4 \cdot n^{3.45+\log(\lceil m/\log(n)\rceil+2)})$$

where the bound cited here stems from Stephan's teaching material [60, Theorem 20.22] while the research papers [22,33,43] obtained slightly better bounds due to some assumptions they make on the game and due to the usage of better bounds for binomials. However, the main contribution of the subsequent research [22,43] is that the quasipolynomial time algorithm can be modified such that in addition to the time bound the workspace the algorithm uses is only quasilinear in the number of nodes n. This improves over the here presented algorithm which uses quasipolynomial space. Furthermore, various authors provided their own version of the verification of the algorithm presented in this paper [22,33,43]. Before presenting the results, the basic properties of the two games (parity and Muller) is given by the following summary.

1. A *game* is given by a directed finite graph of n nodes, a starting node and a set G of sets of nodes which are called the *winning set* of player Anke.
2. Two players, Anke and Boris, move alternately a marker through the graph, starting from some starting node and each time along an outgoing edge of the current position; without loss of generality, every node has at least one outgoing edge (which can go to the node itself). In certain cases, one also considers games in a framework where the current node and not the turn determines which player moves; both

types of games can be translated into each other in polynomial time and the number of nodes doubles at most.

3. A *play* is the infinite sequence of nodes visited by the marker while Anke and Boris are playing. To decide the *winner of a play*, one considers the set of infinitely often visited nodes U. Now Anke wins the play iff $U \in G$. Both types of games, parity games and coloured Muller games, equip the nodes with additional information (values or colours) and say that G must respect this information. In these cases, G depends only on the values or colours of the members of U itself.
4. For *parity games*, the additional information is that each nodes carries a value from $\{1, 2, \ldots, m\}$ and then G either contains all sets U where the maximal value of a node in U is odd (in this case, Anke has odd parity and Boris has even parity) or contains all sets U where the maximal value of a node in U is even (in this case, Anke has even parity and Boris has odd parity).
5. For *coloured Muller games*, each node has a colour and the colours of set U of nodes is the set of all colours which are taken by some nodes in U. Furthermore, G respects the colours in the sense that for every set $U \in G$ and every set U' of nodes having the same colours as U, it holds that also $U' \in G$. So one could represent G also as the set of all combinations of colours where Anke wins. An equivalent model is to permit multiple colours (including none) per node, which is sometimes more succinct and clearer to write; if such a game has m colours and n nodes, it can be translated in polynomial time into an equivalent standard game, that is, into a coloured Muller game with same winner which has exactly one colour per node, where the overall number of colours is $m+1$ and number of nodes is $n \cdot (m+1)$.
6. The number m of colours used in the game is an important parameter of coloured Muller games; for complexity-theoretic considerations, the exact complexity class of solving coloured Muller games with n nodes and m colours might also depend on how G is represented, in particular in the case that m is large, the size of that formula (which in turn might depend on the way it is represented) is a further parameter. However, this parameter is not studied in the present work.
7. A *strategy for a player*, say for Anke, maps, for every situation where Anke has to move, the current position and history of previous moves to a suggested move for Anke. A *winning strategy* for Anke is a strategy for Anke which guarantees that Anke wins a play whenever

she follows the suggested moves. A strategy is called *memoryless* iff it only depends on the current node and not on any other aspects of the history of the play.

8. The *winner of a game* is that player who has a winning strategy for this game. Parity games and Muller games always have a winner and this winner wins every play in the case that the winner follows a winning strategy.

2 The Complexity of the Parity Game

The main result in this section is an alternating polylogarithmic space algorithm to decide the winner in parity games; later more concrete bounds will be shown. The idea is to collect for both players in the game, Anke and Boris, in polylogarithmic space statistics over their performance in the play: these statistics store information about whether the play has gone through a loop where the largest valued node has the parity of the corresponding player.

The following notation will be used throughout the paper. In order to avoid problems with fractional numbers and $\log(0)$, let $\lceil \log(k) \rceil = \min\{h \in \mathbb{N} : 2^h \geq k\}$. Furthermore, a function (or sequence) f is called *increasing* whenever for all i, j the implication $i \leq j \Rightarrow f(i) \leq f(j)$ holds.

Theorem 1. *There exists an alternating polylogarithmic space algorithm deciding which player has a winning strategy in a given parity game. When the game has n nodes and the values of the nodes are in the set $\{1, 2, \ldots, m\}$, then the algorithm runs in $O(\log(n) \cdot \log(m))$ alternating space.*

Proof. The idea of the proof is that, in each play of the parity game, one maintains winning statistics for both players Anke and Boris. These statistics are updated after every move for both players. In case a player plays according to a memoryless winning strategy for the parity game, the winning statistics of this player will eventually indicate the win (in this case one says that the "winning statistics of the player mature") while the opponent's winning statistics will never mature. This will be explained in more detail below.

The winning statistics of Anke (Boris) has the following goal: to track whether the play goes through a loop where the largest value of a node in the loop is of Anke's (Boris') parity. Note that if Anke follows a memoryless winning strategy then the play will eventually go through a loop and the node with the largest value occurring in any loop the play goes

through is always a node of Anke's parity. Otherwise, Boris can repeat a loop with the largest value being of Boris' parity infinitely often and thus win, contradicting that Anke is using a memoryless winning strategy.

The naive method to do the tracking is to archive the last $2n + 1$ nodes visited, to find two identical moves out of the same node by the same player and to check whose parity has the largest value between these two moves. This would determine the winner in the case that the winner uses a memoryless winning strategy. This tracking needs $O(n \cdot \log(n))$ space – too much space for the intended result. To save space one constructs a winning statistics which still leads to an Anke win in case Anke plays a memoryless winning strategy, but memorises only partial information.

The winning statistics of the players are used to track whether certain sequences of nodes have been visited in the play so far and the largest value of a node visited at the end or after the sequence is recorded. The definitions are similar for both players. For simplicity the definition is given here just for player Anke.

Definition 2. In Anke's winning statistics, an i-sequence is a sequence (not necessarily consecutive, but in order) of nodes $a_1, a_2, , a_3, , \ldots, a_{2^i}$ which has been visited during the play so far such that, for each $k \in \{1, 2, 3, \ldots, 2^i - 1\}$, the maximum value of the nodes visited between a_k and a_{k+1}, that is,

$$\max\{val(a) : a = a_k \vee a = a_{k+1} \vee a \text{ was visited between } a_k \text{ and } a_{k+1}\},$$

is of Anke's parity.

The aim of Anke is to find a sequence of length at least $2n + 1$, as such a sequence must contain a loop. So she aims for an $(\lceil \log(n) \rceil + 2)$-sequence to occur in her winning statistics. Such a sequence is built by combining smaller sequences over time in the winning statistics.

Here a winning statistics $(b_0, b_1, \ldots, b_{\lceil \log(n) \rceil + 2})$ of a player consists of $\lceil \log(n) \rceil + 3$ numbers between 0 and m where $b_i = 0$ indicates that currently no i-sequence is being tracked and $b_i > 0$ indicates that

> **Property-b_i**: an i-sequence is being tracked and that the largest value of a node visited at the end of or after this i-sequence is b_i.

Note that for each i at most one i-sequence is tracked. The value b_i is the only information of an i-sequence which is kept in the winning statistics.

The following invariants are kept throughout the play and are formulated for Anke's winning statistics; those for Boris' winning statistics are

defined with the names of Anke and Boris interchanged. In the description below, "i-sequence" always refers to the i-sequence being tracked in the winning statistics.

(I1) Only b_i with $0 \leq i \leq \lceil \log(n) \rceil + 2$ are considered and each such b_i is either zero or a value of a node which occurs in the play so far.
(I2) An entry b_i refers to an i-sequence which occurred in the play so far iff $b_i > 0$.
(I3) If b_i, b_j are both non-zero and $i < j$ then $b_i \leq b_j$.
(I4) If b_i, b_j are both non-zero and $i < j$, then in the play of the game, the i-sequence starts only after a node with value b_j was visited at or after the end of the j-sequence.

When a play starts, the winning statistics for both players are initialised with $b_i = 0$ for all i. During the play when a player moves to a node with value b, the winning statistics of Anke is updated as follows – the same algorithm is used for Boris with the names of the players interchanged everywhere.

1. If b is of Anke's parity or $b > b_i > 0$ for some i, then one selects the largest i such that
   (a) either b_i is not of Anke's parity – that is, it is either 0 or of Boris's parity – but all b_j with $j < i$ and also b are of Anke's parity
   (b) or $0 < b_i < b$

   and then one updates $b_i = b$ and $b_j = 0$ for all $j < i$.
2. If this update produces a non-zero b_i for any i with $2^i > 2n$ then the play terminates with Anke being declared winner.

Note that it is possible that both 1.(a) and 1.(b) apply to the same largest i. In that case, it does not matter which case is chosen, as the updated winning statistics is the same for both cases. However, the tracked i-sequences referred to may be different; this does not effect the rest of the proof.

Example 3. Here is an example of i-sequences for player Anke. This example is only for illustrating how the i-sequences and b_i's work; in particular this example does not use memoryless strategy for any of the players. Consider a game where there is an edge from every node to every node (including itself) and the nodes are $\{1, 2, \ldots, 7\}$ and have the same values as names; Anke has odd parity. Consider the following initial part of a play:

```
1 6 7 5 1 4 5 3 2 1 3 2 3 1 3 3 1 2 1
```

The i-sequences and the b_i's change over the course of above play as given in the following table. In the table, the nodes prefixed by "i :" are the nodes of the corresponding i-sequence.

Move	b_4, b_3, b_2, b_1, b_0	i-sequences in play so far	rule
1	0,0,0,0,1	0:1	1.(a)
6	0,0,0,0,6	0:1 6	1.(b)
7	0,0,0,0,7	1 6 0:7	1.(a)
5	0,0,0,5,0	1 6 1:7 1:5	1.(a)
1	0,0,0,5,1	1 6 1:7 1:5 0:1	1.(a)
4	0,0,0,5,4	1 6 1:7 1:5 0:1 4	1.(b)
5	0,0,0,5,5	1 6 1:7 1:5 1 4 0:5	1.(a)
3	0,0,3,0,0	1 6 2:7 2:5 1 4 2:5 2:3	1.(a)
2	0,0,3,0,0	1 6 2:7 2:5 1 4 2:5 2:3 2	
1	0,0,3,0,1	1 6 2:7 2:5 1 4 2:5 2:3 2 0:1	1.(a)
3	0,0,3,3,0	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3	1.(a)
2	0,0,3,3,0	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3 2	
3	0,0,3,3,3	1 6 2:7 2:5 1 4 2:5 2:3 2 1:1 1:3 2 0:3	1.(a)
1	0,1,0,0,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1	1.(a)
3	0,3,0,0,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3	1.(b)
3	0,3,0,0,3	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 0:3	1.(a)
1	0,3,0,1,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1	1.(a)
2	0,3,0,2,0	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1 2	1.(b)
1	0,3,0,2,1	1 6 3:7 3:5 1 4 3:5 3:3 2 3:1 3:3 2 3:3 3:1 3 1:3 1:1 2 0:1	1.(a)

If at an update of an i-sequence both possible updates 1.(a) and 1.(b) apply to the same level i then it does not matter for the statistics which is chosen. However, for the i-sequences, one has to commit to one choice and for simplicity, (for the above table) one assumes that 1.(a) has priority. So the formal algorithm for updating the sequences is the following one.

1. If b is of Anke's parity or $b > b_i > 0$ for some i, then one selects the largest i such that
   (a) either b_i is not of Anke's parity – that is, it is either 0 or of Boris's parity – but all b_j with $j < i$ and also b are of Anke's parity
   (b) or $0 < b_i < b$.

   else there is no update and one goes to step 3.
2. For the selected i, one does the following update according to the first of the two above cases which applies:
   (a) Let $b_i = b$ and let the i-sequence contain all the nodes of the j-sequences with $j < i$ in that order plus the new node with value b and let $b_j = 0$ for all $j < i$ as the corresponding j-sequences are merged into the new i-sequence;

(b) Let $b_i = b$ and let the i-sequence be unchanged except for the update of the associated value b_i. Furthermore, all j-sequences with $j < i$ are made void by setting $b_j = 0$ for all $j < i$.

Furthermore, all j-sequences with $j > i$ are maintained as they are.

3. If this update produces a non-zero b_i for any i with $2^i > 2n$ then the play terminates with Anke being declared winner and no further tracking of i-sequences is needed.

The 3-sequence in the above table has already a loop, as there are three occurrences of "3 : 3" and the second and third of these have that the same player moves. However, as the sequences are not stored but only the b_i, Anke's winning statistics only surely indicates a win of player Anke when there is an $i > \log(2n + 1)$ with $b_i > 0$; this i is 4 as $2^4 > 2 \cdot 7 + 1$.

Verification of the algorithm. Note that, in the updating algorithm for Anke's winning statistics, if b is of Anke's parity, then there is an i that satisfies 1.(a), as otherwise the algorithm would have terminated earlier. Initially, the invariants clearly hold as all b_i's are 0. Now it is shown that the invariants are preserved at updates of the b_i's according to cases 1.(a) or 1.(b).

It is easy to verify that the invariants are maintained if the update is done due to 1.(b), and it also ensures that Property-b_i is maintained for the i-sequences being tracked. In case the update is done due to 1.(a), then the Property-$b_{i'}$ is maintained for all i'-sequences being tracked for $i' > i$ (with $b_{i'} \geq b$ in these cases). For $i' < i$, $b_{i'}$ is made 0 by the update algorithm. The next paragraph argues about an appropriate i-sequence being formed. Thus, it is easy to verify that (I1) to (I4) are maintained by the update algorithm. Note that (I1) implies that the space bound needed is at most $O(\log n \log m)$, (I2) is used implicitly to indicate which i-sequences are being tracked, and (I3) gives the order of the i-sequences tracked: a $(j + 1)$-sequence appears earlier in the play than j-sequence. This is used implicitly when one combines the smaller j-sequences into a larger one as mentioned below.

When updating Anke's winning statistics by case 1.(a), one forms a new i-sequence of length 2^i by putting the older j-sequences for $j = i - 1, i - 2, \ldots, 1, 0$ together and appending the newly visited one-node sequence with value b; when $i = 0$, one forms a new 0-sequence of length 2^0 consisting of just the newly visited node with value b. Note that in case $i > 0$ both b and b_0 are of Anke's parity and therefore the highest valued node between the last member a of the older 0-sequence and the

last node in the new i-sequence (both inclusive) has the value $\max\{b_0, b\}$ (by (I4) and Property-b_0 for the older 0-sequence). Furthermore, for every $j < i-1$, for the last node a of the older $(j+1)$-sequence and the first node a' of the older j-sequence, in the new i-sequence a highest valued node in the play between these two nodes a, a' (both inclusive) has value b_{j+1} (by (I4) and Property-b_{j+1} of older $(j+1)$-sequence) which, by choice, has Anke's parity. Thus the overall combined new sequence indeed satisfies the properties needed for an i-sequence, b is the value of the last node of this sequence and thus, currently, also the largest value of a node visited at or after the end of the sequence. All older j-sequences with $j < i$ are discarded and thus their entries are set back to $b_j = 0$.

The same rules apply to the updates of Boris' winning statistics with the roles of Anke and Boris interchanged everywhere.

Claim 4. *If a player is declared a winner by the algorithm, then the play contains a loop with its maximum valued node being a node of the player.*

To prove the claim, it is assumed without loss of generality that Anke is declared the winner by the algorithm. The play is won by an i-sequence being observed in Anke's winning statistics with $2^i > 2n$; thus some node occurs at least three times in the i-sequence and there are $h, \ell \in \{1, 2, 3, \ldots, 2^i\}$ with $h < \ell$ such that the same player moves at a_h and a_ℓ and furthermore $a_h = a_\ell$ with respect to the nodes $a_1, a_2, a_3, \ldots, a_{2^i}$ of the observed i-sequence. The maximum value b' of a node between a_h and a_ℓ in the play is occurring between some a_k and a_{k+1} (both inclusive) for a k with $h \leq k < \ell$. Now, by the definition of an i-sequence, b' has Anke's parity. Thus a loop has been observed for which the maximum value of a node in the loop has Anke's parity.

Claim 5. *If a player follows a memoryless winning strategy, then the opponent is never declared a winner.*

To prove the claim, suppose that a player follows a memoryless winning strategy but the opponent is declared a winner. Then the opponent, by Claim 4, goes into a loop with the maximum node of the opponent's parity. Hence, the opponent can cycle in that loop forever and win the play, a contradiction.

Claim 6. *If a player follows a memoryless winning strategy then the player is eventually declared a winner.*

To prove the claim, it is assumed that the player is Anke, as the case of Boris is symmetric. The values b_i analysed below refer to Anke's winning

statistics. Assume that the sequence of values of the nodes in an infinite play of the game has the limit superior c which, by assumption, is a value of Anke's parity. To prove the claim one needs to argue that eventually $b_{\lceil \log n \rceil + 2}$ becomes non-zero. For this purpose it will be argued that some counter associated with the values of b_i's eventually keeps increasing (except for some initial part of the play, where it may oscillate). This is argued by using $count(c, t)$ below, which gives the value of the counter after t steps of the play.

Consider a step as making a move and updating of the statistics. For each step t let $b_k(t)$ refer to the value of b_k at the end of step t (that is, after the updates in the statistics following the t-th move in the play). Let $B_c(t)$ be the set of all k such that $b_k(t)$ has Anke's parity and $b_k(t) \geq c$. Now define

$$count(c, t) = \sum_{k \in B_c(t)} 2^k$$

Now it is shown that whenever at steps t, t' with $t < t'$, a move to node with value c was made and no move, strictly between steps t, t', was made to any node with value $c' \geq c$, then $count(c, t) < count(c, t')$. To see this, let i be the largest index for which there is a step t'' with $t < t'' \leq t'$ such that b_i is updated at step t''.

Note that this implies [$b_i(t) < c$ or $b_i(t)$ is of Boris's parity], and [$0 < b_i(t'') \leq c$]. Now, in the case that $b_i(t'') < c$, it holds that $t'' < t'$ and at time t', condition 1.(b) of the update algorithm will ensure that an update (either 1.(a) or 1.(b)) is done to enforce $b_i(t') = c$. Thus

$$count(c, t') - count(c, t) \geq 2^i - \sum_{j \in B_c(t): j < i} 2^j \geq 1.$$

Accordingly, once all moves involving nodes larger than c in value have been done in the play, there will still be infinitely many moves to nodes of value c and for each two subsequent such moves at t, t' the inequality $count(c, t) + 1 \leq count(c, t')$ will hold. Consequently, the number $count(c, t)$, for sufficiently large t where a move to a node with value c is made at step t, rely on some i with $b_i(t) \geq c$ and $2^i > 2n$ and later the termination condition of Anke will terminate the play with a win.

The above arguments show that a alternating Turing machine can simulate both players and, taking the winning statistics into account, will accept the computation whenever Anke has a winning strategy for the game. Besides the winning statistics, the alternating Turing Machine only needs to keep track of the current node in the play and the player who is to move next. Thus, the alternating Turing machine uses only $O(\log(n) \cdot \log(m))$

space to decide whether the parity game, from some given starting point, will be won by Anke (or Boris), provided the winner plays a memoryless winning strategy (which always exists when the player can win the parity game). □

Chandra, Kozen and Stockmeyer [11] showed how to simulate an alternating Turing machine working in polylogarithmic space by a deterministic Turing machine working in quasipolynomial time. Their simulation bounds for the alternating Turing machine described in Theorem 1 give a deterministic Turing machine working in time $O(n^{c\log(m)})$ for some constant c. As mentioned above, one can always assume that in a parity game with n nodes, with values from $\{1, 2, \ldots, m\}$, one can choose $m \leq n$, so one gets the following parameterised version of the main results that parity games can be solved in quasipolynomial time.

Theorem 7. *There is an algorithm which finds the winner of a parity game in time $O(n^{c\log(m)})$ for a parity game with n nodes and values from $\{1, 2, \ldots, m\}$, with $m \leq n$.*

For some special choices of m with respect to n, one can obtain even a polynomial time bound. McNaughton [48] showed that for every constant m, one can solve a parity game with n nodes having values from $\{1, 2, \ldots, m\}$ in time polynomial in n; however, in all prior works the degree of this polynomial depends on m [31]; subsequent improvements were made to bring the dependence from approximately $n^{m+O(1)}$ first down to $n^{m/2+O(1)}$ [8,59] and then to approximately $n^{m/3+O(1)}$ [42,58]. The following theorem shows that one can bound the computation time by a fixed-degree polynomial in n, for all pairs (m, n) with $m < \log(n)$.

Theorem 8. *If $m < \log(n)$ then one can solve the parity game with n nodes having values from $\{1, 2, \ldots, m\}$ in time $O(n^5)$.*

Proof. Note that Theorem 1 actually shows that the following conditions are equivalent:

- Anke can win the parity game.
- Anke can play the parity game such that her winning statistics matures while Boris' winning statistics does not mature.

Thus one can simplify the second condition and show that it is equivalent to the following two games [46,60]:

- One only maintains Anke's winning statistics and a play terminates with a win for Anke iff she is eventually declared a winner and the play is a win for Boris iff it runs forever.
- One only maintains Boris' winning statistics and a play is a win for Anke iff it never happens that the winning statistics of Boris make him to be declared a winner.

The first game is called a reachability game [46] and the second game a survival game [60, Chapter 9]. Both games are isomorphic, as they are obtained from each other only by switching the player who is supposed to win. Such type of reductions, though not with good complexity bounds, were also considered by Bernet, Janin and Walukiewicz [2]. The reachability game to which one reduces the parity game, can now be described as follows.

- The set Q of nodes of the reachability game consists of nodes of the form $(a, p, \tilde{b})$ where a is a node of the parity game, the player $p \in \{\text{Anke}, \text{Boris}\}$ moves next and $\tilde{b}$ represents the winning statistics of Anke.
- The starting node is $(s, p, \tilde{0})$, where $\tilde{0}$ is the vector of all b_i with value 0, s is the starting node of the parity game and p is the player who moves first.
- Anke can move from $(a, \text{Anke}, \tilde{b})$ to $(a', \text{Boris}, \tilde{b}')$ iff she can move from a to a' in the parity game and this move causes Anke's winning statistics to be updated from $\tilde{b}$ to $\tilde{b}'$ and $\tilde{b}$ does not yet indicate a win of Anke.
- Boris can move from $(a, \text{Boris}, \tilde{b})$ to $(a', \text{Anke}, \tilde{b}')$ iff he can move from a to a' in the parity game and this move causes Anke's winning statistics to be updated from $\tilde{b}$ to $\tilde{b}'$ and $\tilde{b}$ does not yet indicate a win of Anke.

The number of elements of Q can be bounded by $O(n^4)$. First note that the number of increasing functions from $\{0, 1, \ldots, \lceil \log(n) \rceil + 2\}$ to $\{1, 2, \ldots, \lceil \log(n) \rceil\}$ can be bounded by $O(n^2)$, as any such sequence $(b'_0, b'_1, \ldots, b'_{\lceil \log(n) \rceil + 2})$ can be represented by the subset $\{b'_k + k : 0 \leq k \leq \lceil \log(n) \rceil + 2\}$ of $\{1, 2, \ldots, 2\lceil \log(n) \rceil + 2\}$ and that there are at most $O(n^2)$ such sets. Further, note that $b'_k \leq b'_{k+1}$ implies $b'_k + k < b'_{k+1} + k + 1$ and thus all b'_k can be reconstructed from the set. Given a winning statistics $\tilde{b} = (b_0, b_1, \ldots, b_{\lceil \log(n) \rceil + 2})$, one defines $b'_0 = \max\{1, b_0\}$ and $b'_{k+1} = \max\{b'_k, b_{k+1}\}$ and notes that only those b_k with $b_k = 0$ differ from b'_k. Thus one needs $\lceil \log(n) \rceil + 3$ additional bits to indicate which b_k is 0. The overall winning statistics can then be represented by $3\lceil \log(n) \rceil + 5$

bits. Furthermore, one needs 1 bit to represent the player and $\lceil \log(n) \rceil$ bits to represent the current node in the play. Accordingly, each node in Q can be represented with $4\lceil \log(n) \rceil + 6$ bits resulting in $O(n^4)$ nodes in Q. The set Q itself can be represented by using a set of such representations of nodes.

The reachability game can be decided in time $O(|Q| \cdot n)$ by a well-known algorithm. For the general case of a reachability game, the time complexity is linear in the number of vertices plus number of edges of the game graph. The algorithm is listed explicitly by Khaliq and Imran [45] and appeared much earlier in the literature, though in other or only related settings [1,16,37,39,34]. The algorithm is now included for the reader's convenience.

First, one can construct the set Q of vertices and determine a list of nodes $Q' \subseteq Q$ where Anke's winning statistics indicate a win in time $O(|Q| \cdot n)$; the set Q' is the set of target nodes in the reachability game.

Second, one computes for each node $q \in Q$, a linked list of q's successors (which are at most n in number) and a linked list of q's predecessors. Note that the collection of all the successor and predecessor lists for different nodes in Q taken together has the length at most $|Q| \cdot n$. These lists can also be generated in time $O(|Q| \cdot n)$.

Note that a node q is a winning node for Anke if $q \in Q'$ or either Anke moves from q and one successor node of q is a winning node for Anke or Boris moves from q and all successor nodes of q are winning node for Anke. This idea leads to the algorithm below.

Next, for each node q, a tracking number k_q is introduced and maintained such that the winning nodes for Anke will eventually all have $k_q = 0$, where k_q indicates how many further times one has to visit the node until it can be declared a winning node for Anke. The numbers k_q are initialised by the following rule:

- On nodes $q \in Q'$ the number k_q is 1;
- On nodes $q = (a, \text{Anke}, \tilde{b}) \notin Q'$, the number k_q is initialised as 1;
- On nodes $q = (a, \text{Boris}, \tilde{b}) \notin Q'$, the number k_q is initialised as the number of nodes q' such that Boris can move from q to q'.

These numbers can be computed from the length of the list of successors of q, for each $q \in Q$. Now one calls the following recursive procedure, initially for all $q \in Q'$ such that each call updates the number k_q. The recursive call does the following:

- If $k_q = 0$ then return without any further action else update $k_q = k_q - 1$.

- If after this update it still holds $k_q > 0$, then return without further action.
- Otherwise, that is when k_q originally was 1 when entering the call, recursively call all predecessors q' of q with the same recursive call.

After the termination of all these recursive calls, one looks at k_q for the start node q of the reachability game. If $k_q = 0$ then Anke wins else Boris wins.

In the above algorithm, the predecessors of each node $q \in Q$ are only called at most once, namely when k_q goes down from 1 to 0; furthermore, this is the time where it is determined that the node is a winning node for Anke. Thus there are at most $O(|Q| \cdot n)$ recursive calls and the overall complexity is $O(|Q| \cdot n)$.

For the verification, the main invariant is that, for nodes $q \in Q - Q'$, k_q indicates how many more successors of q one still has to find which are winning nodes for Anke until q can be declared a winning node for Anke. In case that Anke's winning statistics has matured in the node q, the value k_q is taken to be 1 so that the node is processed once in all the recursive calls in the algorithm. For nodes where it is Anke's turn to move, only one outgoing move which produces a win for Anke is needed. Consequently, one initialises k_q to 1 and as soon as this outgoing node is found, k_q goes to 0, which means that the node is declared a winning node for Anke. In case the node q is a node where Boris moves then one has to enforce that Boris has no choice but to go to a winning node for Anke. Thus k_q is initialised to the number of moves which Boris can make in this node; each time when one of these successor nodes is declared a winning node for Anke, k_q goes down by one. Observe that once the algorithm is completed, the nodes with $k_q = 0$ are exactly the winning nodes for Anke in the reachability game. □

This special case shows that, for each constant m, the parity game with n nodes having values from $\{1, 2, \ldots, m\}$ can be solved in time $O(n^5)+g(m)$, for some function g depending only on m, and the constant in $O(n^5)$ being independent of m. Such problems are called fixed parameter tractable (**FPT**), as for each fixed parameter m the corresponding algorithm runs in polynomial time and this polynomial is the same for all m, except for the additive constant $g(m)$ depending only on m.

The main complexity classes for a problem of size n (for example nodes in a game) and a parameter m (for example the number of values in a parity game or the number of colours in a coloured Muller game)

are **FPT** of problems which can be solved in time $f(m) \cdot n^k$ or time $g(m) + n^{k+1}$ for some constant k and functions f, g and **XP** of problems which can be solved in time $O(n^{f(m)})$ for some function f. Between **FPT** at the bottom and **XP** at the top are the levels of the **W**-hierarchy **W[1]**, **W[2]**, ...; it is known that **FPT** is a proper subclass of **XP** and it is widely believed that all these levels of the hierarchy are different. Note that the **NP**-complete problems are spread out over all the levels of this hierarchy and that even the bottom level **FPT** also contains sets outside **NP**. The level of a problem can depend on the choice of the parameter m to describe the problem, therefore one has to justify the choice of the parameter m which, in the present work, is a very natural parameter of the problem (number of values or colours of the game). The interested reader is referred to the books of Downey and Fellows [20,21] and Flum and Grohe [25] for further information on parameterised complexity.

The next result carries over the methods of Theorem 8 to the general case, that is, use everything except those parts which make use of $\log(m) \leq n$. So the size of the code representing a winning statistics for Anke is given by $\lceil \log(n) \rceil + 3 \leq \log(n) + 4$ numbers of $\lceil \log(m+1) \rceil \leq \log(m) + 1$ bits. As $\log(m) \leq \log(n)$, the overall size of represention of a node in the set Q of nodes of the reachability game can be bounded by $\log(n) \cdot (\log(m) + 5) + c$. Hence, the size of $|Q|$ is $O(n^{\log(m)+5})$ and the number of edges in the reachability game is $O(n^{\log(m)+6})$.

One can combine this with the usual repeated tests for various types of **NP**-problems: for finding the winning strategy for a player which has a winning strategy, say Anke, in the parity game on graph (V, E), one keeps doing the following:

1. One maintains, for each node $a \in V$, a list of possible successors V_a which is initialised as $\{b : (a, b) \in E\}$ at the beginning.
2. If there is no node $a \in V$ with, currently, $|V_a| > 1$,
   then one terminates with a winning strategy for Anke in the parity game being to move from every node a to the unique node in V_a,
   else one selects a node $a \in V$ with $|V_a| > 1$.
3. Now one splits V_a into two nearly equal sized subsets V'_a and V''_a with $|V'_a| \leq |V''_a| \leq |V'_a| + 1$.
4. One replaces V_a by V'_a and permits, in the derived reachability game, moves from $(a, Anke, \tilde{b})$ to $(a', Boris, \tilde{b}')$ only when $a' \in V_a$.
5. If Anke does not win this game, then one replaces $V_a = V''_a$; else one keeps $V_a = V'_a$.
6. Go to step 2.

The above algorithm works since whenever Anke has a winning strategy for the parity game, then there is a memoryless one and therefore when splitting the options at node a, some memoryless winning strategy either always takes a node from V'_a or always takes a node from V''_a. It is straightforward to verify that the above loop runs $n\log(n)$ rounds and each round involves $O(|Q| \cdot n)$ time plus one solving of the reachability game, which can also be solved in time $O(|Q| \cdot n)$. Thus one can derive the following result.

Theorem 9. *There is an algorithm which finds the winner of a parity game in time $O(n^{\log(m)+6})$ for a parity game with n nodes and values from $\{1, 2, \ldots, m\}$. Furthermore, the algorithm can compute a memoryless winning strategy for the winner in time $O(n^{\log(m)+7} \cdot \log(n))$.*

Follow-up work obtained better bounds on the run-time by using that the translation into the reachability game provides a game with at most

$$\binom{m + 2 \cdot (\lceil \log(n) \rceil + 3)}{\lceil \log(n) \rceil + 3} \cdot n^2$$

edges. This led to the bound $O(2^m \cdot n^4)$ [12] which is based on the fact that $\binom{i}{j} \leq 2^i$ for all i, j. A further estimate can be obtained by slightly increasing the binomial upper bound to

$$\binom{(\lceil m/\log(n) \rceil + 2) \cdot (\lceil \log(n) \rceil + 3)}{\lceil \log(n) \rceil + 3} \cdot n^2$$

and then using common estimates on binomials, where the upper number is a multiple of the lower number. The calculations provide a runtime bound of

$$O(\lceil m/\log(n) \rceil^4 \cdot n^{3.45+\log(\lceil m/\log(n) \rceil+2)});$$

this and similar bounds of this type were obtained by several researchers [22,33,43,60] and subsequent improvements included replacing the term n^2 in the above formulas by the number of edges in the parity game [22,33,43].

The main improvement over the current algorithm by follow-up work is, however, the usage of space. The current algorithm uses quasipolynomial time and quasipolynomial space. Subsequent work has brought down this complexity from quasipolynomial to quasilinear [22,43].

3 Parity Games versus Muller Games

Muller games are a well-studied topic [6,7,48,62,64] and they had been investigated as a general case already before researchers aimed for the more specific parity games. A Muller game (V, E, s, G) consists of a directed graph (V, E), a starting node s and a set $G \subseteq \{0,1\}^V$. For every infinite play starting in s, one determines the set U of nodes visited infinitely often during the play: if $U \in G$ then Anke wins the play else Boris wins the play. In a Muller game the complement of G is closed under union iff for all $U, U' \notin G$, the set $(U \cup U')$ is not in G.

For complexity assumptions, it is natural to consider the case where G is not given as an explicit list, but as a polynomially sized polynomial time algorithm computing the membership of a set U (given by its explicit list) in the set G or some similar equivalent effective representation. The reason for considering such a representation for G is that Horn [36] showed that if G is given as an explicit list of all possible sets of nodes infinitely visited when Anke wins, then the resulting game is solvable in polynomial time in the sum of the number of nodes and the number of explicitly listed sets. Hence, only more flexible ways of formulating winning conditions permit to cover interesting cases of Muller games.

For Muller games, Björklund, Sandberg and Vorobyov [4] considered a parameter which is given by the number of colours. For this, they assign to every node a value or colour from $\{1, 2, \ldots, m\}$ and take G to be some set of subsets of $\{1, 2, \ldots, m\}$. Then U is not the set of infinitely often visited nodes, but instead, the set of colours of the infinitely often visited nodes. Again, if $U \in G$, then Anke wins the play, else Boris wins the play. Coloured Muller games permit more compact representations of the winning conditions. In the worst case there is a 2^m-bit vector, where m is the number of colours; however, one also considers the case where this compressed winning condition is given in a more compact form, say by a polynomial sized algorithm or formula.

In the following, the interactions between Muller games, memoryless winning strategies and parity games are presented. The first result is due to Zielonka [64, Corollary 11] and the second one is in the thesis of Hunter [38].

Theorem 10 (Zielonka [64]). *Given a Muller game (V, E, s, G), assume that the complement of the set G of winning conditions is closed under union. Now, if Anke has a winning strategy in this Muller game then Anke has also a memoryless winning strategy.*

Proof. The possible choices for Anke at any node will be progressively constrained. The proof is by induction on the number of possible moves of Anke in the constrained game. The result holds when, for each node, Anke has only one choice of move. For the induction step, suppose some node v for Anke's move has more than one choice. It is now shown that for some fixed Anke's move at node v, Anke has a winning strategy; thus one can constrain the move of Anke at node v and by induction this case is done. Suppose, by way of contradiction, that for every Anke's move w at v, Boris has a winning strategy S_w. This allows Boris to have a winning strategy for the whole game as follows.

Assume without loss of generality that the play starts with Anke's move at v. Intuitively, think of Boris playing several parallel plays against Anke (each play in which Anke moves w at node v, for different values of w) which are interleaved. For ease of notation, consider the individual play with Anke using move w at node v as play H_w, and the interleaved full play as H.

Initially H and all the plays H_w, are at the starting point. At any time in the play H, if it is Anke's move at v and Anke makes the move w', then Boris continues as if it is playing the play $H_{w'}$ (and suspends the previous play H_w if $w \neq w'$). Thus the nodes visited in H can be seen as the merger of the nodes visited in the plays H_w, for each choice w of Anke at node v. This implies that the set of nodes visited infinitely often in H is equal to the union of the sets of nodes visited infinitely often in the various H_w. As Boris wins each play H_w which is played for infinitely many moves, by closure of the complement of G under union, Boris wins the play H. □

As a parity game is also a Muller game in which G is closed under union for both Anke and Boris, the following corollary holds.

Corollary 11 (Emerson and Jutla [28], Mostowski [49]). *The winners in parity games have memoryless winning strategies.*

Hunter [38, page 23] found a characterisation for Muller games. Note that McNaughton [48] also investigated Muller games with memoryless strategies and characterised them through the concept of splitting [48], which is just another way of stating that both G and its complement are union-closed. However, his paper does not connect these Muller games with parity games explicitly.

Theorem 12 (Hunter [38]). *Every Muller game (V, E, s, G) in which both G and its complement are closed under the union operation is a*

parity game and the translation can be done in polynomial time whenever the winning set G can be decided in polynomial time.

Proof. In this proof a parity game isomorphic to the given Muller game will be constructed. In this parity game player Anke owns the nodes with even value and Boris owns the nodes with odd value. Given V, let

$$V_1 = \{a \in V : \{a\} \in G\} \text{ and } V_2 = \{b \in V : \{b\} \notin G\}.$$

Obviously V is the disjoint union of V_1 and V_2. By the closure under union, any subset $V' \subseteq V_1$ is in G and no subset $V' \subseteq V_2$ is in G.

To prove the theorem, values will be inductively assigned to the nodes one by one.

Suppose values have already been assigned to all nodes in $V - V'$, where V' is initially V. Then, assign the value to one node in V' as follows. Let $V_1' = V' \cap V_1$ and $V_2' = V' \cap V_2$.

Case 1: Suppose $V' \in G$. Then, there is a node $a \in V_1'$ such that $\{a\} \cup V_2' \in G$; otherwise, $V' \notin G$ since complement of G is closed under the union operation. Now let $V_1'' \subseteq V_1'$ and $V_2'' \subseteq V_2'$. The set $\{a\} \cup V_2''$ is in G, as otherwise $(\{a\} \cup V_2'') \cup V_2'$ is not in G, in contradiction to the choice of a. Furthermore, as $V_1'' \cup \{a\} \in G$, $(V_1'' \cup \{a\}) \cup (\{a\} \cup V_2'') = \{a\} \cup V_1'' \cup V_2''$ is in G. Thus whenever $V'' \subseteq V'$ and $a \in V''$, $V'' \in G$. Hence, the value $2|V'|$ is assigned to a accordingly.

Case 2: Suppose $V' \notin G$. Then, there exists a node $b \in V_2'$ such that $\{b\} \cup V_1' \notin G$, by reasons similar to those given in Case 1. Note that this implies that whenever $V'' \subseteq V'$ and $b \in V''$ then $V'' \notin G$. Hence, the value $2|V'| + 1$ is assigned to b.

The above process of assigning values to nodes is clearly consistent, as in Case 1, if a is in V'' then Anke wins and in Case 2, if b is in V'' then Boris wins. It follows that this Muller game is a parity game. □

Besides the standard coloured Muller game of Björklund, Sandberg and Vorobyov [4], one can also consider the memoryless coloured Muller game. These are considered in order to see whether the game is easier to solve if one permits Anke only to win when she follows a memoryless strategy, otherwise she looses by the rules of the game. The main finding is that while memoryless coloured Muller games are, on one hand, easier in terms of the complexity class to which they belong, and, on the other hand, their time complexity is worse and one cannot exploit small numbers of colours as well as in Muller games. This is the main message of the following lines.

Björklund, Sandberg and Vorobyov [4] proved that the coloured Muller game is fixed-parameter tractable iff the parity game is fixed-parameter tractable (with respect to the number of values m of the parity

game). It follows from Theorem 8 that also the coloured Muller game is fixed-parameter tractable.

More precisely, McNaughton [48] and Björklund, Sandberg and Vorobyov [4] showed that a coloured Muller game with m colours and n nodes can be translated into a parity game with $2m$ colours and $m! \cdot n$ nodes. Note that $\log(m! \cdot n) \geq 2m$ for all $m \geq 24$ and $n \geq m$: $\log(m!) \geq \log(8^{m-8}) \geq 3 \cdot (m-8) = 3m - 24$. For $m \geq 24$, $3m - 24 \geq 2m$. Thus, the remaining cases can be reduced to finite ones by observing that for all m and $n \geq \max\{m, 2^{48}\}$, $\log(m! \cdot n) \geq 2m$. So, for almost all pairs of (m, n), $\log(m! \cdot n) \geq 2m$ and therefore one can use the polynomial time algorithm of Theorem 8 to get the following explicit bounds.

Theorem 13. *One can decide in time $O(m^{5m} \cdot n^5)$ which player has a winning strategy in a coloured Muller game with m colours and n nodes.*

For the special case of $m = \log(n)$, the corresponding number of nodes in the translated parity game is approximately $n^{\log(\log(n))+2}$ and the polynomial time algorithm of Theorem 8 becomes an $O(n^{5 \log\log(n)+10})$ algorithm. The algorithm is good for this special case, but the problem is in general hard and the algorithm is slow.

Rabin games and Streett games are games where the winning set G of the game is determined by a list of pairs of sets of nodes $(A_1, B_1), (A_2, B_2), \ldots, (A_m, B_m)$. Now, in the Rabin case, Anke wins a play iff there is an i such that the set of infinitely often visited nodes U intersects A_i and is disjoint to B_i, in the Streett case, Anke wins a play iff all i satisfy that either U intersects A_i or is disjoint to B_i.

Rabin games and Streett games can be translated into Muller games by doubling up the number of colours, that is an n-node Rabin game or Streett game with m conditions can be translated into an n-node Muller game and $2m$ colours. The game graph remains exactly the same and each node caries between 0 and $2m$ colours, more precisely for each of the pairs (A_i, B_i) of the Rabin condition, a node v carries the colour a_i if $v \in A_i$ and the colour b_i if $v \in B_i$. Player Anke wins a play in the Rabin game iff the set U of infinitely often visited nodes satisfies that, for some i, some node in U has colour a_i while no node in U has colour b_i. The winning condition for the Streett game is complementary.

Now one can decide in time $O((2m)^{10m} \cdot n^5)$ which player is the winner of the Rabin game or Streett game. Alternatively, one can use the direct translation of Rabin games and Streett games to parity games [4,48], which they used to show that these games are in **FPT** iff parity games are. This translation increases the number of nodes by a factor $m! \cdot m^2$

and the colours by a factor 2 and gives better bounds for the final algorithm of approximately $O(m^{5m} \cdot n^5)$, which is the same bound as the Muller game has. A direct solution without translating into other games by Piterman and Pnueli [53] has the runtime $O(n^{m+1} \cdot m!)$.

One might ask whether this bound can be improved. Björklund, Sandberg and Vorobyov [4] showed that under the Exponential Time Hypothesis it is impossible to improve the above algorithm to $O(2^{o(m)} \cdot n^c)$, for any constant c. Here the Exponential Time Hypothesis says that the problem **3SAT** with n variables is not solvable in time $O(2^{o(n)})$. The following result enables to get a slightly better lower bound based on the more likely assumption that $\mathbf{FPT} \neq \mathbf{W[1]}$. Here a dominating set of a graph is a set of nodes such that from every node in the graph there is an edge to one of the nodes in the dominating set; for this property one deviates from the usual convention of the non-existence of self-loops and assumes that every node has a loop to itself.

Theorem 14. *Given a graph H with n nodes and a number m with the constraint that $2 \leq m \leq n$, one can compute in time polynomial in n' a coloured Muller game with n' nodes and m' colours such that,*

- $m' \leq (6m \cdot \log(n)/\log(m \cdot \log(n)))$,
- $n' \leq 2^{(4m \cdot \log(n)/\log(m \cdot \log(n)))} \cdot m \cdot n \cdot (2m \cdot \log(n)/\log(m \cdot \log(n)))$ *and*
- *the given graph H has a dominating set of size up to m iff player Anke has a winning strategy in the resulting coloured Muller game.*

Proof. Assume that the given graph H has vertices $\{a_1, \ldots, a_n\}$ and let E be the set of its edges. Without loss of generality assume that $n, m \geq 2$ so that $\log(n), \log(m)$ are at least 1.

Note that a dominating set can be represented using $m \cdot \lceil \log(n) \rceil$ bits, where each node is represented using $\lceil \log(n) \rceil$ bits. Let o be the smallest natural number such that $o \geq 2$ and $\log(o)$ is an integer and $o \cdot \log(o) \geq m \cdot \lceil \log(n) \rceil$.

The game constructed below uses a set C of $2o$ colours and a set D of o colours. Intuitively, in each "round" of the play (consisting of several moves):

- Boris chooses a vertex $v \in V$ (except in the first round where v is some default).
- Anke needs to show that some neighbouring vertex w of v is h-th member (for $1 \leq h \leq m$) in the fixed dominating set chosen by Anke.
- The choice of Anke's dominating set $\{a_{k_1}, a_{k_2}, \ldots, a_{k_m}\}$ (represented using bits $b_1 b_2 \ldots b_{m \cdot \lceil \log(n) \rceil}$) is expressed in the play by Anke choosing

o times a new member of C to form a set $\tilde{C}$ of size o (note that the sequence of the o choices is important along with the set $\tilde{C}$ itself).

- Representing each choice by $\log(o)$ bits gives the $o\log(o)$ bits needed for representing the dominating set chosen by Anke. These choices of Anke must be consistent with w being the h-th member of the dominating set.

The next problem to be addressed is to make sure that Anke's choice of the dominating set is consistent for different values of v chosen by Boris. This is ensured by allowing Boris to "break" the rounds in case the choices made by Anke are inconsistent. How this is done will become more clear in the proof below. Now the formal construction is given.

The game is now described below, by giving the names of the nodes and the possible moves. The colour of a node is the fourth component of the tuple used as node-name. The overall colours used are $D = \{d_1, \ldots, d_o\}$ and a further set C of $2o$ colours disjoint to D. Although the players move alternately, it is always the same player in each node which moves. Note that the chosen dominating set of Anke is represented using $m \cdot \lceil \log(n) \rceil \leq o \log o$ bits. In the following, for checking if the dominating set chosen by Anke is consistent with the choice of neighbour (w, h), it is checked if the bits $b_{(h-1)\lceil \log n \rceil + 1} b_{(h-1)\lceil \log n \rceil + 2} \cdots b_{h\lceil \log n \rceil}$ denote the node w. As Anke choose $\log o$ bits at a time, the above consistency check may need to be done over several choices of colours. For this, let "choice (r, ℓ) is consistent with (w, h)" mean the following condition: the binary representations $d_1 d_2 \ldots d_{\log(o)}$ of $\ell - 1$ and $w_1 w_2 \ldots w_{\lceil \log(n) \rceil}$ of w satisfy that for all i, j with $1 \leq i \leq \log(o)$ and $1 \leq j \leq \lceil \log(n) \rceil$, if $r \cdot \log(o) + i = (h-1) \cdot \lceil \log(n) \rceil + j$ then $d_i = w_j$.

1. Nodes $\{(v, \emptyset, 1, d_1) : v \in V\}$. These nodes have colour d_1 and Anke moves in these nodes. Any fixed node among these modes can be considered as a start node. There is an edge from (v) to $(w, \{e\}, h, e)$ (described in item 2 below), if $(v, w) \in E$, $1 \leq h \leq m$, c is the ℓ-th colour of C with $1 \leq \ell \leq o$ and the choice $(0, \ell)$ is consistent with (w, h).
   Intuitively, node $(v, \emptyset, 1, d_1)$ represents for Anke the task to test whether there is a neighbour of v in the dominating set chosen by Anke and in her move she choses w and says that it is the h-th node in her list; the choice of the first colour c describing this list must be consistent with the h-th element being w.
2. Nodes $\{(w, \tilde{C}, h, c) : w \in V, 1 \leq h \leq m$ and $\tilde{C} \subseteq C$ and $1 \leq |\tilde{C}| \leq o\}$ and $c \in \tilde{C}$. Boris moves in these nodes.

In this situation, Anke has just added c to the colours of $\tilde{C}$ and one knows what the encoded last $\log(o)$ bits are, as they correspond to the position of c in $(C - \tilde{C}) \cup \{c\}$. In the case that $|\tilde{C}| < o$, Boris can either ask Anke to continue or present Anke with a new task; in the case that $|\tilde{C}| = o$, Boris has to present a new task; the case $|\tilde{C}| > o$ does not occur.
For presenting a new task, Boris moves to $(v, \emptyset, 1, d_1)$ for some vertex v of H; for asking Anke to go on, Boris moves to $(w, \tilde{C}, h, d_{|\tilde{C}|+1})$.

3. Nodes $\{(w, \tilde{C}, h, d_{|\tilde{C}|+1}) : w \in V,\ 1 \leq h \leq m \text{ and } \tilde{C} \subseteq C \text{ and } 1 \leq |\tilde{C}| < o\}$.
In this situation, $|\tilde{C}| \cdot \log(o)$ of the bits describing the dominating set have been processed and now Anke selects an $\ell \in \{1, 2, \ldots, o\}$ such that the choice $(|\tilde{C}|, \ell)$ is consistent with (w, h) and now Anke can move to $(w, \tilde{C} \cup \{c\}, h, c)$ where c is the ℓ-th element of $C - \tilde{C}$.

Now, the winning condition for Boris is as follows: there is a number r such that exactly the first r colours in D are on the nodes visited infinitely often while at least $r + 1$ of the colours in C are on nodes visited infinitely often. It is easy to see that these winning conditions are closed under union and therefore Anke, if she can win, has a memoryless winning strategy.

Now consider any play in the game and assume that Anke plays a memoryless strategy – it is sufficient to show how to counter those. Consider a round in the play as moves of the play going from one visit of a node of the form $(v, \emptyset, 1, d_1)$ to the next visit of the node of the form $(v', \emptyset, 1, d_1)$. Then, up to the time when Boris decides to go to $(v', \emptyset, 1, d_1)$, the set $\tilde{C}$ has grown to some size r and each of its colours was the colour of a node moved to by Anke and each of $d_1, d_2, \ldots, d_r$ was the colour of a node moved to by Boris (or a start node). So a subset $\tilde{C}$ of r colours plus the r colours $d_1, d_2, \ldots, d_r$ occurred on the way. In the case that Anke is inconsistent, there are a least r and two nodes v, v' such that when coming from $(v, \emptyset, 1, d_1)$ and $(v', \emptyset, 1, d_1)$ two different sets $\tilde{C}$ and $\tilde{C}'$ of r colours in C are covered and so Boris' strategy would be to go alternately to $(v, \emptyset, 1, d_1)$ and $(v', \emptyset, 1, d_1)$ and then to follow Anke's unwinding of the bits of the dominating set for r double-moves and then to start a new round. This would result in the set $\tilde{C} \cup \tilde{C}' \cup \{d_1, \ldots, d_r\}$ of colours being visited infinitely often and as $|\tilde{C} \cup \tilde{C}'| > r$ it would be a win for Boris.

In the case that there is no dominating set of size m, then Anke will assign for two different vertices v, v' two different nodes w, w' with the same parameter h and then Boris can exploit this, as this would force Anke to become inconsistent at some r; so Boris would have the above

method to counter a hypothetical memoryless winning strategy of Anke.

Now suppose there is a dominating set of size m, then Anke can fix such a set $(a_{k_1}, a_{k_2}, a_{k_3}, \ldots, a_{k_m})$ and the corresponding coding $b = b_1 b_2 b_3 \ldots b_{m \cdot \lceil \log(n) \rceil}$ of it. Now, from a node $(v, \emptyset, 1, d_1)$, Anke always chooses $(a_{k_h}, \{c\}, h, c)$, where a_{k_h} is a neighbour of v and the ℓ-th colour c of C where $\ell - 1$ has a binary representation which agrees with the first $\log(o)$ bits of b. From a node $(a_{k_h}, \tilde{C}, h, d_{|\tilde{C}|})$ with $|\tilde{C}| < o$, Anke would choose the ℓ-th colour c from $C - \tilde{C}$ and move to $(a_{k_h}, \tilde{C} \cup \{c\}, h, c)$ where c is the ℓ-th colour of $C - \tilde{C}$ and ℓ is chosen such that the binary representation of $\ell - 1$ equals to the $\log(o)$ bits of b starting with the $|\tilde{C}| \cdot \log(o)$ plus first bit of b. The goal of this is that Anke chooses the colours always such that the current (r, ℓ) is consistent with all pairs $(a_{k_1}, 1)$, $(a_{k_2}, 2)$, $(a_{k_3}, 3)$, $\ldots$, (a_{k_m}, m). Thus her choices will be through-out consistent and it is easy to see that this is a winning strategy for Anke.

In summary, Anke is the winner of the so constructed game iff the graph H has an dominating set of size m.

The constructed game has $3o$ colours. Furthermore, the first coordinate of a node $(w, \tilde{C}, h, c)$ is a node, the second is a subset of C of size up to o, the third coordinate is a number from 1 to m and the fourth coordinate is either an element of $\tilde{C}$ or $d_{|\tilde{C}|+1}$ where the latter case does not occur when $|\tilde{C}| = o$. Thus one can see that the first coordinate has n choices, the second has at most 2^{2o} choices, the third has m choices and the fourth has at most o choices. So there are at most $2^{2o} \cdot n \cdot o \cdot m$ nodes. As o is strictly bounded by $2m \cdot \log(n)/\log(m \cdot log(n))$ (for large enough value of $m \log(n)$), the overall number of colours is strictly bounded by $6m \cdot \log(n)/\log(m \cdot \log(n))$, and corresponding the number of nodes is bounded by $2^{4m \cdot \log(n)/\log(m \cdot log(n))} \cdot m \cdot n \cdot (2m \cdot \log(n)/\log(m \cdot log(n)))$. □

If the Muller game would now have an algorithm which runs in time $2^{o(m' \cdot \log(m'))} \cdot (n')^c$ for some constant c then the overall runtime would be of order $2^{o(m \cdot \log(n))} \cdot (2^{m \cdot \log(n)/\log(m \cdot \log(n))} \cdot n^4)^c$. One can bound the term $2^{o(m \cdot \log(n))}$ by $2^{o(m) \cdot \log(n)}$ which in turn is $n^{o(m)}$. Furthermore, one can bound the term $(2^{m \cdot \log(n)/\log(m \cdot \log(n))})^c$ by $(n^{o(m)})^c$ which in turn is again $n^{o(m)}$. So one obtains that the overall runtime is be estimated as $n^{o(m)} \cdot n^{4c+2}$ which would be $n^{o(m)}$ as the constant $4c+2$ is $o(m)$. It would follow that one could check in runtime $n^{o(m)}$ whether there is a bounded dominating set of size m in a graph with n nodes. Chen, Huang, Kanj and Xia [15, Theorem 5.4] showed that if there is an algorithm which solves dominating set problem with these parameters – note that their paper uses a different notation – then $\mathbf{FPT} = \mathbf{W[1]}$. Thus one has the below

corollary. To see its sharpness, note that the **FPT**-algorithm for Muller games obtained in Theorem 13 above has the complexity $O(2^{5m\log(m)} \cdot n^5)$.

Corollary 15. *A Muller game with m colours and n nodes cannot be solved in time $2^{o(m\cdot\log(m))} \cdot n^{O(1)}$ unless* $\mathbf{FPT} = \mathbf{W[1]}$.

The above reduction, due to the optimisation involved, increased massively the number of nodes in the game. If one does not want to lower the factor m to $(m/\log(m)) \cdot \log(n)$ in the translation, but only cares to achieve $O(m \cdot \log(n))$ colours, then there are more straightforward methods. One might ask whether there is a translation which uses less nodes but increases the number of colours massively. Zielonka [64] used similar methods to show **NP**-hardness of the Muller games, even in the special case of games where player Anke, in the case that she wins, also has a memoryless winning strategy.

Theorem 16 (Zielonka [64]). *The problem to determine whether Anke can win a Muller game when the set of Boris' winning conditions is closed under union is* **NP***-complete; however, for containment in* **NP***, the winning conditions have to be represented as in Zielonka's paper. In general, this problem is in* $\mathbf{\Sigma_2^P}$.

Note that for games where Anke might win, but not with a memoryless winning-strategy, the complexity bound is worse. Dawar, Horn and Hunter showed that the problem to decide the winner in a Muller game is **PSPACE**-complete [17].

Memoryless games are games where Anke wins iff she (a) plays a memoryless strategy and (b) wins the game according to the specification of the game. If she does not do (a), this is counted as a loss for her. This was already done by Björklund, Sandberg and Vorobyov [4, Section 5] for Streett games and it can also be done for Muller games.

In contrast to normal coloured Muller games, the complexity of the memoryless games is different. Björklund, Sandberg and Vorobyov [4, Section 5] considered memoryless Streett games (called Quasi-Streett games in their paper) and showed that these are $\mathbf{W}[1]$-hard. This result implies that memoryless coloured Muller games are $\mathbf{W}[1]$-hard. Furthermore, on one hand, one can decide in $\mathbf{\Sigma_2^P}$ whether Anke has a winning strategy: There is a memoryless strategy of Anke such that the graph obtained by fixing Anke's moves according to the strategy, does not allow Boris to reach a loop of length up to $2n^2$, where the set of colours of this loop is a non-member of G. On the other hand, the next result shows that unless **NP** can be solved in quasipolynomial time there is no analogue of the

translation of Björklund, Sandberg and Vorobyov [4] from memoryless coloured Muller games into parity games. In contrast, solving memoryless coloured Muller games with four colours is already **NP**-complete. So, unless $\mathbf{P} = \mathbf{NP}$, the problem is not even in **XP**.

Theorem 17. *Solving memoryless coloured Muller games with four colours is* **NP***-hard.*

Proof. In the following, satisfiability is reduced to memoryless coloured Muller game as follows. For ease of writing the proof, Muller games where nodes determine the player moving are considered. This could be easily converted to a game where the moves of Anke and Boris alternate by inserting intermediate nodes if needed.

Suppose $x_1, x_2, \ldots, x_k$ are the variables and $y_1, y_2, \ldots, y_h$ are the clauses in a satisfiability instance. Without loss of generality assume that no variable appears both as positive and negative literal in the same clause. Then, the above instance of satisfiability is reduced to the following Muller game (where the graph is undirected graph):

1. $V = \{s\} \cup \{u_1, u_2, \ldots, u_k\} \cup \{v_1, v_2, \ldots, v_h\} \cup \{w_{i,j} : [1 \leq i \leq h]$ and $[1 \leq j \leq k]$ and $[x_j$ or $\neg x_j$ appears in the clause $y_i]\}$.
2. $E = \{(v_i, w_{i,j}), (w_{i,j}, x_j) : x_j$ or $\neg x_j$ appears in $y_i\} \cup \{(s, u_i) : 1 \leq i \leq k\}$.
3. The colours are $\{x, y, +, -\}$; s has the colour y, all nodes u_j have the colour x; all nodes v_i have the colour y; for every node $w_{i,j}$ in the graph, if x_j appears in the clause y_i positively then the colour is $+$ else $\neg x_j$ appears in y_i and the colour is $-$.
4. The winning sets for Boris are $\{x, +, -\}$ and all subsets of $\{y, +, -\}$; the winning sets for Anke are $\{x, +\}$, $\{x, -\}$, $\{x\}$ and all supersets of $\{x, y\}$.

Now it is shown that the instance of satisfiability problem is satisfiable iff Muller game is a win for Anke playing in a memoryless way.

Suppose the instance is satisfiable. Then fix a satisfying assignment $f(x_j)$ for the variables, and let $g(y_i) = j$ such that x_j (or $\neg x_j$) makes the clause y_i true. Now Anke has the following winning strategy: At node v_i, move to $w_{i,g(y_i)}$. At node $w_{i,j}$, if $g(y_i) = j$ then move to u_j else move to v_i. Intuitively, at nodes v_i, Anke directs the play to the node $u_{g(y_i)}$ (via $w_{i,g(y_i)}$). Similarly, for the nodes $w_{i,j}$, Anke directs the play to $u_{g(y_i)}$ either directly or via nodes v_i and $w_{i,g(y_i)}$.

Thus, clearly, if an infinite play goes through colour y infinitely often, then it also goes through colour x infinitely often; thus Anke wins. On

the other hand, if an infinite play does not go through colour y infinitely often, then the set of nodes the play goes through infinitely often is, for some fixed j, u_j and some of the nodes of the form $w_{i,j}$. But then, by the definition of Anke's strategy, the play can only go through nodes of colour $-$ finitely often (if $f(x_j)$ is true) and through nodes of colour $+$ finitely often (if $f(x_j)$ is false). Thus, Anke wins the play.

Now suppose Anke has a winning strategy. If there is an i such that Anke moves from $w_{i,j}$ to u_j then let $f(x_j)$ be true if x_j appears positively in clause y_i; else let $f(x_j)$ be false. If there is no i such that Anke moves from $w_{i,j}$ to u_j then truth value of $f(x_j)$ does not matter (and can assigned either true of false).

To see that above is a satisfying assignment, first note that for each clause y_i, there exists a $w_{i,j}$ such that Anke moves from $w_{i,j}$ to u_j. Otherwise, Boris can first move from start node to u_j and then to $w_{i,j}$ such that x_j appears in clause y_i; afterwards the play will go infinitely often only through a subset of the nodes of the form $v_i, w_{i,j}$ and thus the colours which appear infinitely often in the above play is a subset of $\{y, +, -\}$.

Furthermore, for no j and two nodes $w_{i,j}$ and $w_{i',j}$ such that x_j appears in y_i and $\neg x_j$ appears in $y_{i'}$, does Anke move from $w_{i,j}$ and $w_{i',j}$ to node u_j. Otherwise, Boris could win by first moving from s to u_j and then alternately going to nodes $w_{i,j}$ and $w_{i',j}$. It follows that f gives a satisfying assignment for the instance of satisfiability. □

4 Conclusion

The progress reported in this paper shows that solving parity games is not as difficult as it was widely believed. Indeed, parity games can be solved in quasipolynomial time – the previous bounds were roughly $n^{O(\sqrt{n})}$ – and they are fixed parameter tractable with respect to the number m of values (aka colours or priorities) – the previously known algorithms were roughly $O(n^{m/3})$. These results are in agreement with earlier results stating that parity games can be solved in **UP** ∩ **co-UP** [41] and that there are subexponential algorithms to solve the problem [44]. In spite of the current progress, *the original question, as asked by Emerson and Jutla [28] in 1991 and others, whether parity games can be decided in polynomial time still remains an important open question.*

The above results on parity games are then used to give an algorithm of runtime $O((m^m \cdot n)^5)$ for coloured Muller games with n nodes and m colours; this upper bound is almost optimal, since an algorithm with runtime $O((2^m \cdot n)^c)$, for some constant c, only exists in the case that

FPT = **W[1]**, an assumption which is considered to be unlikely.

One might ask whether the results obtained for parity games permit further transfers to Muller games, for example, in the special cases where (a) player Anke can employ a memoryless winning strategy due to the special type of the game or (b) one does not permit player Anke to use other strategies than memoryless ones. Note that case (b) differs from case (a), as in case (b) the condition on using memoryless strategies can be restrictive while case (a) applies to Muller games of those types where one knows that "if Anke has a winning strategy then she has a memoryless winning strategy". Case (a) was analysed by McNaughton [48] and Zielonka [64]; it applies to Muller games where the winning condition of player Boris is closed under union [64].

The above mentioned lower bound directly also applies to case (a). For case (b), the complexity class of the general problem is also in the polynomial hierarchy but not **PSPACE**-complete (unless **PSPACE** = $\mathbf{\Sigma}_2^P$) as the decision problem for coloured Muller games; however, the algorithmic bounds are much worse, as one can code **NP**-hard problems into instances with four colours.

ACKNOWLEDGEMENTS. The authors would like to thank Krishnendu Chatterjee, Sasha Rubin, Sven Schewe and Moshe Vardi for correspondence and comments. Further thanks go to the referees of STOC 2017 for numerous suggestions.

References

1. Catriel Beeri. On the membership problem for functional and multivalued dependencies in relational databases. *ACM Transactions on Database Systems*, 5:241–259, 1980.
2. Julien Bernet, David Janin and Igor Walukiewicz. Permissive strategies: from parity games to safety games. *RAIRO - Theoretical Informatics and Applications*, EDP Sciences, 36:251–275, 2002.
3. Dietmar Berwanger and Erich Grädel. Fixed-point logics and solitaire games. *Theory of Computing Systems*, 37(6):675–694, 2004.
4. Henrik Björklund, Sven Sandberg and Sergei Vorobyov. *On fixed-parameter complexity of infinite games.* Technical report 2003-038, Department of Information Technology, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden.
5. Henrik Björklund, Sven Sandberg and Sergei Vorobyov. Memoryless determinacy of parity and mean payoff games: a simple proof. *Theoretical Computer Science*, 310(1–3):365–378, 2004.
6. Hans L. Bodlaender, Michael J. Dinneen and Bakhadyr Khoussainov. On game-theoretic models of networks. *Algorithms and Computation*, Twelfth International Symposium, ISAAC 2001, Christchurch, New Zealand, December 2001, Proceedings. *Springer LNCS*, 2223:550–561, 2001.

7. Hans L. Bodlaender, Michael J. Dinneen and Bakhadyr Khoussainov. Relaxed Update and Partition Network Games. *Fundamenta Informaticae*, 49(4):301–312, 2002.
8. Anca Browne, Edmund M. Clarke, Somesh Jha, David E. Long and Wilfredo R. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theoretical Computer Science*, 178(1–2):237–255, 1997.
9. Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li and Frank Stephan. Deciding parity games in quasipolynomial time. *STOC 2017 Theory Fest: Forty Ninth Annual ACM Symposium on the Theory of Computing*, 19–23 June 2017, Proceedings, ACM, Montreal, Canada, 12 pages, 2017.
10. Felix Canavoi, Erich Grdel and Roman Rabinovich. The discrete strategy improvement algorithm for parity games and complexity measures for directed graphs. *Theoretical Computer Science*, 560:235–250, 2014.
11. Ashok K. Chandra, Dexter C. Kozen and Larry J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.
12. Krishnendu Chatterjee. *Comments on the Quasipolynomial Time Algorithm.* Private communication, 2017.
13. Krishnendu Chatterjee and Thomas A. Henzinger. Strategy Improvement and Randomized Subexponential Algorithms for Stochastic Parity Games. *Twentythird Annual Suymposium on Theoretical Aspects of Computer Science*, STACS 2006, Marseille, France, 23–25 February 2006, Proceedings. *Springer LNCS*, 3885:512–523, 2006.
14. Krishnendu Chatterjee, Marcin Jurdzinski and Thomas A. Henzinger. Quantitative stochastic parity games. SODA 2004, *Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 121–130, 2004.
15. Jianer Chen, Xiuzhen Huang, Iyad A. Kanj and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006.
16. Stephen A. Cook. Path systems and language recognition. *Proceedings of the Second Annual ACM Symposium on Theory of Computing*, STOC 1970, May 4–6, 1970, Northampton, Massachusetts, USA, pp. 70–72, 1970.
17. Anuj Dawar, Florian Horn and Paul Hunter. *Complexity Bounds for Muller Games.* Manuscript, 2011.
18. Antonio Di Stasio, Aniello Murano, Giuseppe Perelli and Moshe Y. Vardi. Solving parity games using an automata-based algorithm. *Twenty first International Conference on Implementation and Application of Automata*, CIAA 2016, 19–22 July 2016, Seoul, South Korea, *Springer LNCS*, 9705:64–76, 2016.
19. Christoph Dittmann, Stephan Kreutzer and Alexandru I. Tomescu. Graph operations on parity games and polynomial-time algorithms. *Theoretical Computer Science*, 614: 97–108, 2016.
20. Rodney G. Downey and Michael R. Fellows. *Parameterised Complexity.* Springer, Heidelberg, 1999.
21. Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity Theory.* Springer, Heidelberg, 2013.
22. John Fearnly, Sanjay Jain, Sven Schewe, Frank Stephan and Dominik Wojtczak. *An Ordered Approach to Solving Parity Games in Quasi Polynomial Time and Quasi Linear Space.* Technical report on `http://arxiv.org/abs/1703.01296`, 2017.
23. Olivier Finkel and Stevo Todorčević. The isomorphism relation between tree-automatic structures. *Central European Journal of Mathematics*, 8(2):299–313, 2010.

24. Olivier Finkel and Stevo Todorčević. A hierarchy of tree-automatic structures. *The Journal of Symbolic Logic*, 77(1):350–368, 2012.
25. Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.
26. Oliver Friedmann. An exponential lower bound for the parity game strategy improvement algorithm as we know it. *Logic in Computer Science*, LICS 2009, pp. 145–156, 2009.
27. E. Allen Emerson. Automata, tableaux, and temporal logics. *Proceedings of the Workshop on Logic of Programs*, *Springer LNCS*, 193:79–88, 1985.
28. E. Allen Emerson and Charanjit S. Jutla. Tree automata, μ-calculus and determinacy. *Annals of IEEE Symposium on Foundations of Computer Science*, pp. 368–377, 1991.
29. E. Allen Emerson, Charanjit S. Jutla, A. Prasad Sistla. On model checking for the μ-calculus and its fragments. *Theoretical Computer Science*, 258(1-2):491–522, 2001.
30. Oliver Friedmann and Martin Lange. Solving parity games in practice. *Automated Technology for Verification and Analysis*, Seventh International Symposium, ATVA 2009, Macao, China, 14–16 October 2009, *Springer LNCS*, 5799:182–196, 2009.
31. Jakub Gajarský, Michael Lampis, Kazuhisa Makino, Valia Mitsou and Sebastian Ordyniak. Parameterized algorithms for parity games. *Mathematical Foundations of Computer Science*, MFCS 2015. *Springer LNCS*, 9235:336–347, 2015.
32. Aniruddh Gandhi, Bakhadyr Khoussainov and Jiamou Liu. Efficient algorithms for games played on trees with back-edges. *Fundamenta Informaticae*, 111(4):391–412, 2011.
33. Hugo Gimbert and Rasmus Ibsen-Jensen. A short proof of correctness of the quasi-polynomial ti8me algorithm for parity games. Technical report on `http://arxiv.org/abs/1702.01953`, 2017.
34. Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema and Scott Weinstein. *Finite Model Theory and Its Applications*. Springer, 2007.
35. Andrey Grinshpun, Pakawat Phalitnonkiat, Sasha Rubin and Andrei Tarfulea. Alternating traps in Muller and parity games. *Theoretical Computer Science*, 521:73–91, 2014.
36. Florian Horn. Explicit Muller games are **PTIME**. *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, FSTTCS 2008, pp. 235–245, *Dagstuhl Technical Reports*, 1756, 2008.
37. Yuri Gurevich and Leo Harrington. Trees, automata and games. *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC 1982, May 5–7, 1982, San Francisco, California, USA, pp. 60-65, 1982.
38. Paul William Hunter. *Complexity and Infinite Games on Finite Graphs*. PhD Thesis, University of Cambridge, Computer Laboratory Hughes Hall, 2007.
39. Neil Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22(3):384–406, 1981.
40. Hajime Ishihara, Bakhadyr Khoussainov. Complexity of some infinite games played on finite graphs. *Graph-Theoretic Concepts in Computer Science*, Twenty-Eighth International Workshop, WG2002, Cesky Krumlov, Czech Republic, 13–15 June 2002, Proceedings. *Springer LNCS* 2573:270–281, 2002.
41. Marcin Jurdzinski. Deciding the winner in parity games is in **UP** $\cap$ **co-UP**. *Information Processing Letters*, 68(3):119–124, 1998.
42. Marcin Jurdziński. Small progress measures for solving parity games. *STACS 2000, Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science*, *Springer LNCS*, 1770:290–301, 2000.

43. Marcin Jurdziński and Ranko Lazić. *Succinct progress measures for solving parity games.* Technical report on `http://arxiv.org/abs/1702.05051`, 2017.
44. Marcin Jurdziński, Mike Paterson and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM Journal on Computing*, 38(4):1519–1532, 2008.
45. Imran Khaliq and Gulshad Imran. Reachability games revisited. *Second International Conference on Advances and Trends in Software Engineering*, SOFTENG 2016, 21–25 February 2016, Lisbon, Portugal, Proceedings, International Academy, Research and Industry Association (IARIA), Wellington, DE 19810, USA, pp. 129–133, 2016.
46. Bakhadyr Khoussainov and Anil Nerode. *Automata Theory and its Applications.* Birkhäuser, 2001.
47. Dietrich Kuske, Jiamou Liu and Markus Lohrey. The isomorphism problem for omega-automatic trees. Proceedings of *Computer Science Logic*, CSL 2010, *Springer LNCS*, 6247:396–410, 2010.
48. Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.
49. Andrzej Wlodzimierz Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdanski, Instytut Matematyki, 1991.
50. Jan Obdrzalek. Algorithmic analysis of parity games. PhD thesis, University of Edinburgh, 2006.
51. Viktor Petersson and Sergei G. Vorobyov. A randomized subexponential algorithm for parity games. *Nordic Journal of Computing*, 8:324–345, 2001.
52. Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3:5):1–21, 2007.
53. Nir Piterman and Amir Pnueli. Faster solutions of Rabin and Streett games. *Twenty First IEEE Symposium on Logic in Computer Science*, LICS 2006, 12–15 August 2006, Seattle, WA, USA, Proceedings. IEEE Computer Society, pages 528–539, 2006.
54. Shmuel Safra. On the complexity of ω-automata. Proceedings twenty-ninth IEEE Symposium on Foundations of Computer Science, pages 319-327, 1988.
55. Shmuel Safra. Exponential determinization for omega-Automata with a strong fairness acceptance condition. *SIAM Journal on Computing*, 36(3):803–814, 2006.
56. Sven Schewe. Solving parity games in big steps. *FCTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, *Springer LNCS*, 4855:449–460, 2007; *Journal of Computer and System Sciences*, available online, 2016.
57. Sven Schewe. From parity and payoff games to linear programming. *Mathematical Foundations of Computer Science 2009*, Thirty-Fourth International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings. *Springer LNCS*, 5734:675–686, 2009.
58. Sven Schewe. Solving parity games in big steps. *Journal of Computer and System Sciences*, 84:243—262, 2017.
59. Helmut Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59:303–308, 1996.
60. Frank Stephan. *Methods and Theory of Automata and Languages.* Lecture Notes, School of Computing, National University of Singapore, 2016. `http://www.comp.nus.edu.sg/~fstephan/fullautomatatheory-nov2016.ps`.
61. Colin Stirling. Bisimulation, modal logic and model checking games. *Logic Journal of IGPL*, 7(1):103–124, 1999.

62. Wolfgang Thomas. On the Synthesis of Strategies in Infinite Games. *Twelfth International Symposium on Theoretical Aspects of Computer Science*, STACS 1995, *Springer LNCS*, 900:1–13, 1995.
63. Thomas Wilkie. Alternating tree automata, parity games and modal μ-calculus. *Bulletin of the Belgian Mathematical Society*, 8(2):359–391, 2001.
64. Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.